

# Developing and Deploying Software Engineering Courseware in an Adaptable Curriculum Framework

**W. Richards Adrion**

University of Massachusetts, Amherst  
Department of Computer Science  
140 Governor's Drive  
University of Massachusetts  
Amherst, MA 01003-4610 USA  
+1 413 545 2475  
adrion@cs.umass.edu

## **ABSTRACT**

We describe an effort to design an *adaptable* framework for teaching and learning in software engineering. We are developing a repository of asynchronous, multimedia courseware that facilitates the rapid incorporation of new advances in research and technology, enables courses to be tailored to individual student needs and interests, leverages innovations in educational technology and encourages innovation in teaching and in student learning. Our emphasis is on developing composable *multi-level* “knowledge and topic units” (KU/TUs) that can be employed to tailor course content and depth to fit the needs of a diverse student population. We have developed “live” and on-line course material for KU/TUs in software engineering and taught courses using this material. The framework was deployed in three software engineering courses (previously taught concurrently) and provides quite different learning environments for the students in each course and, to some extent, tailors the courses to individual students within the classes based on their skills, objectives and backgrounds. We describe efforts at formative evaluation. Student satisfaction is high and available measures of success, e.g., student performance, have improved markedly. We also describe a project now beginning to build on this prototype that will be accompanied by more extensive formative and summative evaluation.

## **Keywords**

Teaching and learning; software engineering education; multimedia courseware; educational technology; pedagogy.

## **1 CHALLENGES AND GOALS**

In the SE academic community, there is little agreement on what should be included in a software engineering curriculum, at what levels it should be taught, whether it belongs as part of a computer science curriculum [13] or whether it should be taught at all [6]. At the same time, the movement towards evaluation and certification, if successful, may define or constrain the “content” of software engineering curricula. These issues clearly arise for other computer science and engineering curricula as well.

In addition to the controversy concerning content of curricula, we are seeing increasing competition from both traditional and non-traditional educational institutions offering distance-learning alternatives and an increasing demand to service students at a time and place and with educational programs of their choosing. Students demand more variety and types of specialization. The shortage of SE and IT professionals results in an increasing demand for continuing education, re-training and certificate programs. A large public university will have to adapt its education delivery systems to remain competitive in today's “marketplace” in education. Leveraging resources (e.g., instructional materials and units) from existing and related courses becomes increasingly attractive. Given the difficulty in determining a single core curriculum, how can we coordinate the increasing variety of curricula demanded in the marketplace, even if there is some agreement on the components of an individual curriculum?

Ours is also a rapidly changing field. With the rapid advances in the field, in SE research and technology, the cost to faculty of maintaining currency outside of our own

specialties is high. We must constantly renew and revise our educational offerings. In a research institution, rewards are given for individual contributions to fairly narrow areas and sub-disciplines, not necessarily broad scholarship. Even if one could keep up with the entire field, the cost of constantly updating courses, materials and curriculum is high. Students and teachers alike have less time, but are required to do more “things.” This means that existing resources (such as classroom and preparation time) must be used as effectively as possible.

Philosophical debates are common in the many research communities concerned with teaching and learning. Educational theorists, cognitive scientists, philosophers, psychologists, instructional designers, human-computer interface designers, etc., are all trying to understand how to facilitate learning. Constructivism is a popular and appealing learning theory with roots in the work of Piaget and Papert. Constructivism suggests that learners actively construct their knowledge, that knowledge cannot merely be transmitted. The companion concept of learner-centered education raises difficult challenges in the traditional university setting, asking us to move from lecturer to mentor. Constructivism and other learning theories are difficult to support solely with instructional technology [21]. Software engineering seems best learned by “doing” and best retained when learned in the context of personal experience.

Thus, we desire to identify new pedagogy (moving from “sage on the stage to guide on the side”); to incorporate learner-centered design in the development of curriculum and instructional strategies; to develop general and flexible curricular frameworks and supporting materials; and to exploit new technologies on-campus and for distance learning. Ideally, we need a malleable curriculum and teaching tools that can be adapted to a particular student population, and yet extensible (by either the student or instructor, individually or together) to suit individual student needs and “marketplace” demands.

Clearly, complicated issues such as those above do not admit an easy, or a single, solution. Before describing our efforts to date, let us first identify the broad goals we seek to achieve (in our computer science curriculum generally, as well as specifically in our software engineering courses):

- *To maintain our curriculum at the cutting edge.* We want to be able to rapidly design degree and certificate programs, develop individual courses and materials, and evolve delivery mechanisms that allow us to keep our educational programs current and relevant.
- *To focus education on fundamental concepts and ideas.* Our curriculum must convey the foundations, for software engineering, that provide a strong knowledge base for a career in an ever-evolving field.
- *To provide students with sufficient depth, breadth and*

*exposure to application domain knowledge.* We need to provide each student with a broad understanding of core concepts, depth in sub-specialty areas, and exposure to application domain knowledge.

- *To provide students with “deep” content.* We want to enable motivated students, or students with interest in a particular area, to explore a topic more deeply on their own, and in the context of the core material they have already learned in that area.
- *To accommodate students with varied preparation and background in flexibly-designed traditional (campus-based) and non-traditional (off-campus) programs.* We are facing the reality that our students are changing. We have an increasing number of non-traditional, off-campus students with the need for flexible time schedules. We are seeing a wide difference in the level of preparation of incoming on-campus students. We need to provide a flexible curriculum on campus to accommodate cooperative programs and work schedules, as well as providing life-long learning opportunities.
- *To effectively employ computing and telecommunications technology.* We want to apply the technologies developed in our research in the courses we offer. To increase our productivity while remaining at the cutting edge of curriculum content, we need to exploit multimedia learning technologies to support and supplement, not supplant our teaching methods.
- *To develop among our faculty the pedagogical skills that will be needed in an environment where multimedia instructional materials are at a student's fingertip.* We are replacing the lecture-oriented mode of “information delivery” with activities involving more active student and instructor participation in the classroom (e.g., group problem solving, discussion/debate).

## 2 DEVELOPING A VISION

For several years we have developed instructional technology to learn what does, and does not, work well by implementation and experimentation. In particular, two projects, MANIC (Multimedia Asynchronous, Networked Instructional Courseware) and OWL (On-line Web-Based Learning), are related to the efforts described herein.

MANIC [8,18,19] is a tool supporting streamed audio/video multimedia Web-based presentations. Unlike many of its competitors [3,4,9,14,22], MANIC has powerful indexing and search tools that allow MANIC presentations to be directly accessed by lecture, topic, slide, bullet and word/character. A student can move immediately to the point in a presentation covering a particular algorithm, for example, and start the audio/video sequencing from that point. The student can also attach private notes to presentation “pages” that are held in a database for later access. These features support learner

problem solving by serving as mechanisms to quickly search for context, structure, and strategies within stored lectures.

The OWL [10] system, initially created to replace an earlier PLATO-based electronic homework system, has grown to be a powerful and discipline independent electronic homework system that includes course management and student record keeping capability. Today, OWL features: delivery via world-wide-web; platform independent authoring, supports several question types (short answer, numerical answer, matching, multiple-multiple choice); has parameterizable input and feedback using ranges and/or tables; and supports text and media tags and tables.

MANIC initially was designed for asynchronous learning, essentially as a powerful replacement for video broadcast and videotape delivery. The MANIC Project team produces on-line materials for short courses and full-length courses using materials (audio, video, instructors notes) from courses that are offered on campus, through campus (Video Instruction Program), federal (DOD DLAMP) and national (National Technical University) distance education, and through the University Arts Extension Service. Currently, most of the courses are publicly available on the WWW. MANIC usage logs indicate that MANIC has delivered hundreds of thousands of pages of notes/audio (more than 4 Billion bytes of data) to more than 25,000 distinct users in more than 100 countries. Many of the courses were converted to MANIC presentations from videotape in their entirety and were indexed by topic and by lecture. OWL is used by over 3700 students in the University's departments of Chemistry, Physics, Geosciences, Education, Entomology, Foreign Languages and Nutrition. Efforts are underway to integrate OWL and MANIC to support the activity described here as well as courses in other departments and through extension.

A few years ago, we began to explore the use of MANIC in our Department's regular course offerings. In conjunction with creating MANIC courseware for a software engineering syllabus that served three courses, traditionally taught concurrently, we experimented with using the MANIC courseware from a new pedagogical viewpoint. The courses serve students with widely varying backgrounds (on- and off-campus students, computer science majors and computer science graduate students, and undergraduate and graduate students from many other disciplines). Instructors for the courses incorporate on-going research results from the Department's Laboratory for Advanced Software Engineering Research (LASER). Approximately 30% of the material "turns over" each year as new research results and new technologies are introduced. As an experiment in Spring 1998, material was extracted from existing MANIC Software Engineering Analysis courseware (used for off-campus VIP/NTU students, but created from on-campus lectures) and was

restructured into a dozen or so groups of topical units. Five or six additional units were recorded outside of class. These units were organized as core, supplemental (e.g., details on standards, particular methods and techniques) and as advanced material (e.g., recent research results from LASER). This allowed the instructor to begin to individualize the material for each of the courses. Following this initial experiment, a more ambitious restructuring was carried out in preparation for the course offered in Spring 1999. This experiment is described below.

### 3 THE KU/TU FRAMEWORK

We needed a framework for organizing the MANIC courseware and experience with the initial experiment led to a framework that builds on the curriculum structure proposed in Curriculum 91[1]. Recognizing the diversity in the field and among students, Curriculum 91 chose not to define *the* computer science curriculum, but rather chose to define a flexible set of so-called "knowledge units"<sup>1</sup> (KUs) to circumscribe core curriculum areas. These KUs in turn can be mixed and matched to tailor courses targeted at a specific student population. The granularity of "knowledge units" and "lecture topics" defined in Curriculum 91 varies, knowledge units are defined in detail for the core material, while advanced and supplemental material is described in a manner similar to typical course syllabi. Knowledge units are not new ideas. In Curriculum 91 and several other curricula, a units are used as a structuring mechanism. Units are common in curriculum and instructional design. In adapting the knowledge unit approach, our intent was to follow the spirit of the Curriculum 91 recommendations in terms of structure, but also to develop our own concepts of appropriate units based on our experience. Our goal in organizing curriculum into knowledge units is to create a framework to facilitate the effective use of educational technology to individualize learning in the classroom. Knowledge units as we will use them serve as a locus for problem-focused and learner-centered instructional strategy. The linkages among units and with their components provide "scaffolding" [16] to assist the learner to structure and personalize that being learned.

We developed the framework shown Figure 1. A course can be thought of as a collection of knowledge units (KUs), and each knowledge unit as a collection of topic units

---

<sup>1</sup> In Curriculum 91, "a knowledge unit is understood to designate a coherent collection of subject matter that is so fundamental ... that it should occur in every undergraduate curriculum. While the subject matter of a knowledge unit may be related to other knowledge units in the common requirements by way of a prerequisite structure, it can nevertheless be introduced within any of several alternative course structures. ... While every knowledge unit is considered to be essential, the depth and breadth of coverage for each topic therein will not be the same."

(TUs). Each topic unit has three levels: *prerequisite content, base content and advanced content*. The

prerequisite content consists of pointers to other topic units (in the same or other knowledge units, or in other courses), remedial units and review units. The base content describes the primary content for the topic unit with pointers to related technology-specific materials (e.g., details on programming languages, operating systems, hardware specifications, laboratory systems, software tools) and, as needed, application-domain-specific or state-of-the-practice-specific units. The advanced content includes pointers to supplemental and advanced units, e.g., other topic units (possibly in other courses or knowledge units), application-domain-specific or state-of-the-practice-specific units (e.g., mechanics, finance, life sciences, standards, product and process specifications) and non-traditional material (e.g., on-line tutorials from professional societies, related campus lectures and colloquia). An example from data flow analysis is given in Figure 1a.

We believe this framework provides two advantages. First, it allows us to identify TUs that can be shared (see Figure 2) across courses and to identify the links between courses at both the knowledge unit and topic unit levels. The units that are shared or to which a substantial number of links exist can be supported by instructional technology (e.g., broadcast, interactive and Web-based video, intelligent tutors, simulations, animations, Web-based materials, links and indices). These technology-supported units can be

maintained in an indexed and searchable repository, where they can be easily accessed by students and instructors. We

feel that it is at this level where technology can be productively and effectively introduced. This level of granularity also offers the best opportunity to facilitate the deployment of technology-enhanced material within the curriculum, since, from our experience, courseware is most likely to be (re-)used at the topic or knowledge unit level. Secondly, we believe that the KU/TU structure can be used to guide curriculum design and development to meet changing needs and demands.

#### 4 EXPERIENCE WITH THE KU/TU APPROACH

In Spring 1997, we created MANIC courseware from the lectures associated with an advanced undergraduate SE course (CMPSCI 521 Software Engineering: Analysis and Evaluation) and an introductory graduate SE course (CMPSCI 621

Advanced Software Engineering: Analysis and Evaluation). As briefly described above, the MANIC courseware was restructured into a KU/TU-like structure in Spring 1998 and additional MANIC units were created outside the classroom. In the initial experiment, CMPSCI 521 was divided into 13 KUs, each with 1-5 TUs. In all, thirty-seven TUs were available as MANIC courseware (and on videotapes). In this preliminary experiment, all but five TUs were covered in regular lecture sessions; the MANIC courseware primarily served as review material for

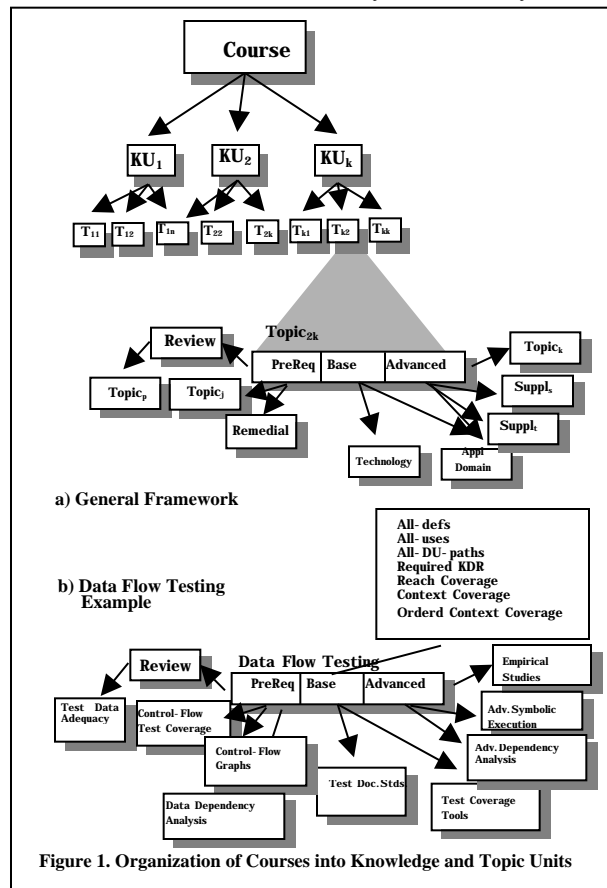


Figure 1. Organization of Courses into Knowledge and Topic Units

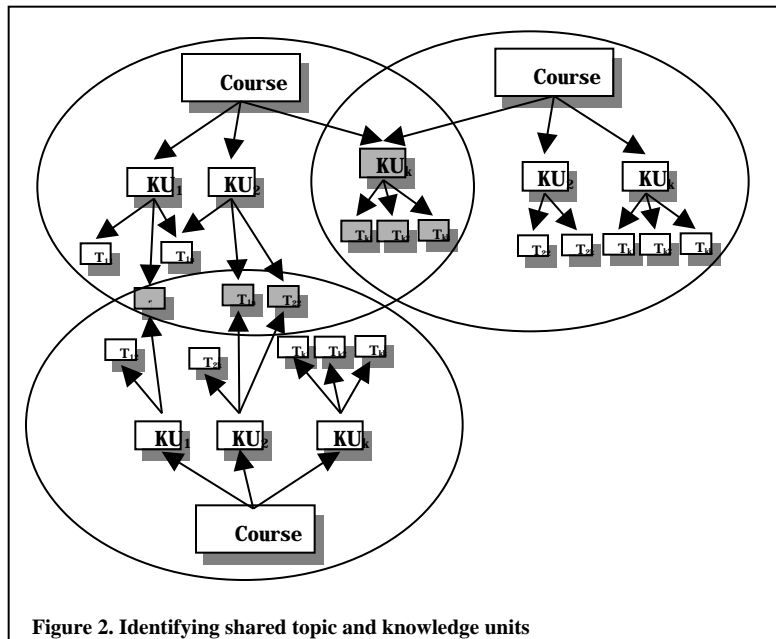


Figure 2. Identifying shared topic and knowledge units

the students. These five additional units served as a first step in individualizing the material for each course.

**Table 1: KU/TU Organization Spring 1999**

KUs	TUs covered in lectures	Supplemental TUs	Advanced TUs
Overview	Overview I Overview II Legal & Other Issues		Foundations
Process & Standards	V&V example Requirements Analysis Source Code Comp. Tools	Process Models IEEE829	
Safety	SW Safety Overview	Safety	
Metrics	Metrics Overview	Metrics-I (req-design), Metrics-II(code)	
Formal Methods	Formal Methods Overview Verification Overview Symbolic Evaluation Overview	Specifications-I Specifications-II Verification I Symbolic Evaluation I	Verification II Symbolic Evaluation-II Social-Processes
Testing	Reviews & Inspections Testing Foundations Testing Strategies I Testing Strategies II Testing O-O Systems I Testing OOS II Applications testing Integration & Regression Testing	Test-Data-Selection-I Test-Data-Selection-II Fault-Based (Mutation) Fault-Based (Relay) Error-Based-Testing Dynamic-Analysis	Dependency-Analysis
Empirical-Studies	Empirical-Studies		
Data flow Analysis	Data flow Analysis Overview	Data-Flow-I Data-Flow-II Interprocedural-DFA	Advanced-DFA
Concurrency	Concurrency	Concurrency-TIG-PN	FLAVERS Conc.-labeled-fg Concurrency-INCA
Software-Arch	Software-Arch-I Soft.-Arch.-II		

Following the initial experiment, we defined a more complete KU/TU structure for the Spring of 1999. Table 1 summarizes the KUs and TUs for the courses (CMPSCI 521, CMPSCI 521 [honors] and CMPSCI 621). All of the TUs were made available on the Web in three formats: full topic-unit text/graphics; converted Powerpoint slides and MANIC presentations. All TUs also were made available on videotape and printed notes in the library (and sent to the off-campus students). A limited number of links to Web reference materials were included on the course home page and in the on-line presentations. We made an effort to identify linkages to other MANIC presentations, specifically compiler, operating system and architecture courses, but time did not allow the linkages to be included in the online presentations.

Of the fifty-one Topic Units available via MANIC, twenty-four units served as review for the same units covered during in-class lectures given concurrently to students in all three courses and twenty-seven were available with instructor narration only on the Web or videotape. The eighteen “Supplemental TUs” were required of all students in each course; the nine “Advanced TUs” were required for students in CMPSCI 621 (with a portion of these also required for CMPSCI 521 [honors]). There were twenty-nine scheduled class meetings. In addition to the covering the TUs, seven (24%) of the class meetings were reserved for “problem-solving” sessions – relatively unstructured sessions where the instructor used example problems to encourage discussion and debate, and to reinforce the formal lectures. The instructor also met extensively with the 621 and honors students outside of class.

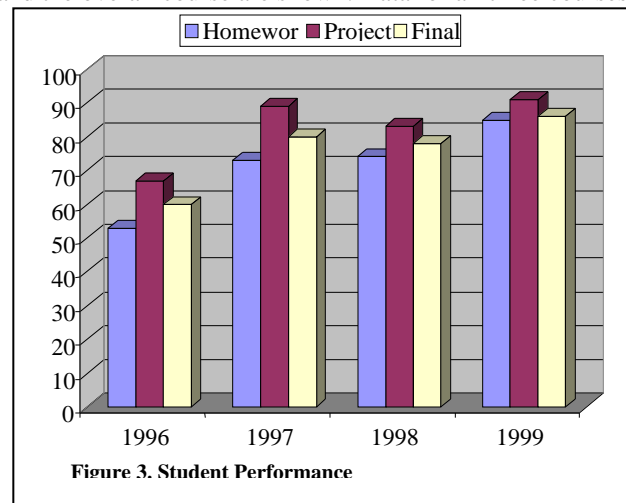
### Initial evaluation

We carried out an analysis of the performance of the students over a four-year period. This period included:

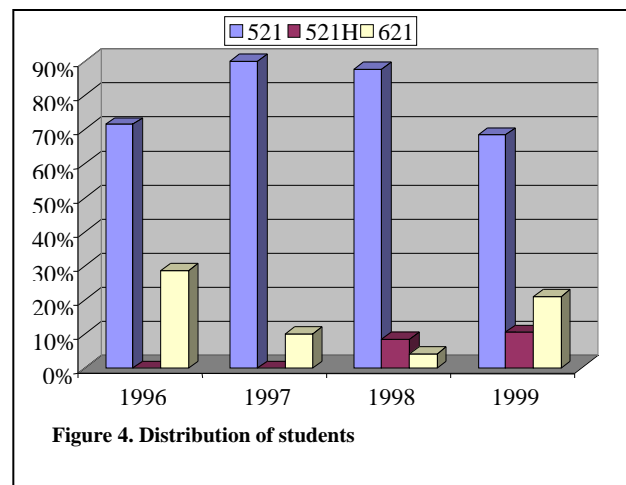
- one year of traditional lectures with all three courses taught concurrently;
- one year, also with the classes combined, where the MANIC versions of the lectures were available within 2-3 days of the actual live lecture for review;
- the initial experiment where a few supplemental lectures (“units”) were available on MANIC that were not covered in class (3 supplemental, 2 advanced); and
- one year where we carried out the more extensive experiment described above.

In 1999, a formative evaluation was carried out through extensive questionnaires given each student in all three classes. In prior years, we had only anecdotal information on student satisfaction with MANIC. We also looked at the course evaluations and carried out analyses of the MANIC logs to see how the MANIC materials were used.

The only readily available measures of student performance are grades. In Figure 3, the grades on homework, projects and the overall course are shown. Data for all three courses



were combined. We did this for two reasons. The courses



initially had been taught as a single course, with differing project requirements and homework assignments (the

additional homework problems for graduate students were factored out of the data); and the distribution of students (see Figure 4) across the courses varied substantially from year.

As you can see from these preliminary data, homework (and overall) grades improved substantially. Not shown here, the distribution of grades also narrowed substantially. These data partially may result from an observed improvement in the overall performance of our undergraduate students. Clearly, the addition of the problem-solving sessions was valuable (and was one of the goals of the framework). The change is dramatic enough, however, to indicate that the framework was effective.

In the first two years where MANIC courseware was available, we found that on-campus students primarily used MANIC materials for review and while working on homework

assignments. This observation is supported by our analysis of the MANIC logs [11] that show students average 45 minutes per MANIC session; 50% used the text-only features (not the audio track); and the average time on play-out was 7 minutes.

Anecdotally, the students found the search/index capability of MANIC particularly useful in locating materials related to specific assignment problems. They also found the online audio material useful in “filling in the gaps” in the associated text and graphical material. The off-campus students often found the MANIC presentations user-friendlier than videotapes because of the index/search capability. On-campus students had some difficulty accessing the Web materials due to load and bandwidth requirements, particularly when accessing the campus via local ISPs. Off-campus students had more difficulty with access from work sites, due to problems with security and firewalls.

The questionnaire completed in Spring 1999 is summarized in Table 2. This is primarily a survey of how the various technologies were used, but also it does provide some insight into the effectiveness of the KU/TU structure. To complete the assignments, the students had to study the

supplemental material (for graduate and honors students that included some or all of the “advanced” material). We provided this material, as noted, in a variety of media. The text, “coursepacks” (copies of articles), and copies of the overhead transparencies (for both “live” and MANIC units) were available to the students. Note that 82% of the students used this material frequently, while 59% used the MANIC units. Of those using the various media, only 29% of the students found the text and coursepack material useful; 88-89% found the copies of transparencies (printed or on-line) useful; and 90% found MANIC useful.

As we found earlier, students who used MANIC used it primarily for review and homework (60-70% used it for these activities), preferring it appears, to first look at the printed material for the supplemental units. A large number of students did note that they found MANIC superior to the printed and on-line Power Point slides, because of the detail

added by the audio track. Several also commented that MANIC was far superior to the videotapes. One student commented that s/he wished that other courses had had as much material available for outside study.

Overall, the students were quite satisfied with the course, indicating that it covered a number of important areas. There were complaints that with all of the

supplemental and advanced material, the course required too much. They did like the “problem solving” sessions and several asked that there be more, perhaps offered by teaching assistants.

Overall attendance did not seem to be impacted too much by availability of all of the in-class lectures on MANIC (and in handouts). While 24% said that class attendance *was* affected by the availability of the on-line material and the handouts, most commented that it provided a level of comfort, not a temptation. Most who indicated that class attendance was affected, also indicated that they attended most of the time. Class attendance seemed to average around 75%, not an atypical figure for a large public institution for a class containing mostly graduating seniors.

As a first full prototype of the framework, we would have to say that it was generally successful. The majority of the students found it valuable and liked the availability of

**Table 2. Questionnaire**

	Most of the time	Once/week	Infrequently	MANIC affect?	
Attended class	59%	24%	18%	24%	
	Text/handout		MANIC	Videotape	
Used	82%	53%	59%	18%	
	Useful	Not useful	% used		
Text/course-pack	29%	65%	47%		
Handouts	88%	12%	100%		
Online Power Point	89%	11%	53%		
MANIC	90%	10%	59%		
Videotape	67%	33%	18%		
	Replace class	Review	Homework	Supplemental	Access Easy?
Used MANIC for	30%	70%	60%	30%	70%
	Office hours	email	Web	MANIC	Video
For questions, used:	50%	44%	65%	53%	12%

printed, videotaped and on-line materials. One lesson is to design the course so as to not overburden students.

## 5 NEXT STEPS

In Fall 1999, we received a three-year grant from the National Science Foundation to continue to develop the KU/TU strategy and supporting on-line units. The primary goal of this research is to improve teaching and learning. The experiment described herein is not particular to software engineering education. SE education seems to benefit from learning by doing or learning in the Constructivism-style. For that reason, SE education may benefit more from this strategy for instruction than other areas of computing. We are motivated to improve our instructional design, but the focus of the experiment and out future works is not on the technology (MANIC, OWL), it is on the *application* of the technology.

We developed two concurrent approaches: a *top-down* effort to organize the Computer Science Department's curriculum to encourage new pedagogy and the use of technology; and a *bottom-up* effort within and across existing courses at the University and other institutions. Our experiences with a small number of courses will serve as additional prototypes for the top-down organization of our overall CS curriculum and as a basis for transferring the work to other institutions. We plan an ongoing evaluation of our efforts at both levels. The bottom-up effort includes the development of Web-based knowledge and topic units and the detailed testing, development and assessment of the KU/TU approach in several University, Williams College, Smith College and Mount Holyoke College courses. We are beginning our top-down approach by undertaking a review of curriculum design efforts on other campuses and within professional societies and a study of our current offerings and strategies with an eye towards identifying their knowledge and topic unit structure

The first phase of the bottom-up effort expands the experiments described above to include most undergraduate-level (at the University and Williams College) software engineering courses and two graduate-level software engineering courses as prototype KU/TU-based courses. In later phases, we will extend the experiment to other multi-campus courses (networking, information retrieval).

We are identifying overlapping knowledge and topic units across all of software engineering courses and identifying links to basic core course or specialized short course knowledge units (algorithms, data structures, discrete mathematics; or Java programming, Unix fundamentals, etc.) that may serve as review, remedial, supplemental and background material. We will revise the topic and knowledge unit structure to maximize the "usability" of shared units. The major thrust of our work will be to develop *content* for the web-based knowledge and topic units. The Web-based knowledge and topic units will be

created using a variety of technologies (MANIC, OWL, Flash, Real Publisher, intelligent tutors, etc.) and by editing and restructuring MANIC courseware to extract "reusable" units. We will organize the Web-based units in an indexed and searchable knowledge base. This will allow us to rapidly incorporate the units to provide depth, breadth, review and remediation in new and existing undergraduate and graduate courses, as stand-alone review and remedial materials for our students; and in short courses for continuing education and lifelong learning. It will allow the highly motivated student to move at a self-determined pace and in a "non-linear" way through curriculum.

Another goal is to incorporate self-assessment to guide students to interrelated topics units across courses. We will aid students in identifying when remediation may be advised, when they can move on to more advanced levels or what "broadening" topics may be of interest. We will integrate the Web-based units into the OWL course management system providing access to the units, testing, homework, grading and administration. Later we will include automated self-testing and finally we will move to built-in student-model-driven advising [14] that will guide the student to related, supplemental and remedial topics.

Third-party evaluation is being incorporated from the onset addressing the quality and effectiveness of the courseware and curriculum, relative to its goals and objectives. Evaluation will integrate qualitative and quantitative measures as a means of providing both meaningful, action-oriented feedback as well as rich documentation of the project and its accomplishments.

Our evaluation process is designed assess whether 1) we are achieving our stated goals and objectives in a timely and effective manner; 2) the modules are easily integrated into courses and successful in replicating the qualities and effectiveness of lecturers; and 3) the modules are easily disseminated and adapted to other sites, and achieve similar results when utilized by faculty other than the authors. We will soon begin summative evaluation to integrate qualitative and quantitative measures as a means of providing both meaningful, feedback and rich documentation of the project and its accomplishments.

Our experience with the design, development and deployment of prototype knowledge and topic units, supported and enhanced by educational technology, in software engineering and in the courses selected in the second and later phases will be essential in guiding the overall top-down computer science curriculum redesign.

## 6 ADDITIONAL OBSERVATIONS

Several observations follow from our experience to date and several issues arose:

- *The cost of analyzing the curriculum to identify KU/TU candidates, the time to prepare instructional materials,*

*implement instructional technology applications and revise courses, and the overall **increase in preparation time***. The level of effort in creating *effective* online content and in creating the additional supplemental material is substantial. MANIC and other streaming media software capture too much of in-class discussions, questions and administrative announcements, but the cost of editing the audio material precluded eliminating these in the initial experiments. Providing sufficient on-line material to support a reasonable level of navigation and to create the scaffolding (context and supporting materials) is quite expensive in terms of instructor time. Linking the materials across courses is underway and seems no more difficult than creating the materials.

- *Is it having an **impact** on education?* Initial review of student performance showed marked improvement. The “no significant difference” phenomenon noticed in other projects [12,15] did not occur here. Students appeared to be satisfied, generally, with the flexibility offered within the KU/TU structure. They did complain about the volume of material that they were asked to cover. The sample space was small and preliminary. Additional and detailed evaluation is needed.
- *Getting faculty to use **materials developed by others***. In courses at the leading edge of technology and research, individual instructors can differ greatly on what should be taught and how it should be taught. Given the substantial investment in creating instructional technology materials, it is incumbent on the designers to produce units that instructors will use. This suggests a process of refinement and redesign that will take some time. It is unlikely that one instructor will be comfortable with using much of what another instructor has created, particularly if it is only slightly edited from a live lecture. Moreover, the time it takes to navigate and evaluate on-line material is probably more than for text material. Experience by our technology transfer partner, ACSIOM [2], indicates that using professional instructional designers, user-interface specialists, voice and presentation “coaches” lead to more professional, effective and usable on-line materials.
- *Access and IP issues*. Currently, many MANIC presentations are publicly available on the Web. We are addressing a number of issues concerning access: should it be limited to those registered for the course through the University? Is the need to receive “credit” or “certification” sufficient to encourage off-campus users to register? Should we allow faculty in other institutions<sup>2</sup> to use our KUs in their courses? Now that

intellectual property is a major topic on campuses, we have to address the ownership of the content in instructional technology modules. Should it be treated as a textbook? As software? How do we manage the use of copyrighted materials in our KUs? In the experiments described above, some courseware draws heavily on materials developed by others. For example, the units on software architecture use examples from the extensive on-line materials at CMU and the SEI. A course under development for the Department of English includes passages from a number of authors. These are not easy issues to address.

## 7 SUMMARY AND CONCLUSIONS

We have described an effort to design an *adaptable* framework for teaching and learning in software engineering. The primary strategy is to incorporate a repository of asynchronous, multimedia courseware that facilitates the rapid incorporation of new advances in research and technology, enables courses to be tailored to individual student needs and interests, leverages innovations in educational technology and encourages innovation in teaching and in student learning. This courseware repository consists of stored on-line multimedia (audio, video, notes), tutoring, and self-testing capabilities constructed using the educational technology developed as part of on-going research in intelligent tutoring and multimedia operating systems and networking.

Our preliminary experience is that student satisfaction is high and that available measures of success, e.g., student performance, have improved markedly as the courseware has been developed and deployed. More evaluation needs to be done. Our current project to expand the experiment will be accompanied by more extensive formative and summative

We believe this strategy represents a significant opportunity to enhance the educational opportunities available to software engineering undergraduates and to explore new teaching skills and paradigms by our faculty.

## ACKNOWLEDGEMENTS

We would like to acknowledge the support of the National Science Foundation through grants EIA-9502639 and EIA-9979833, and the support of the Centers for Real-Time, Intelligent Complex Computing Systems at the University of Massachusetts, Amherst

## REFERENCES

- [1] ACM/IEEE Joint Curriculum Task Force. Computing Curricula 1991 (New York 1991) ACM Press.
- [2] ACSIOM Website Available at <<http://www.acsiom.org/>>
- [3] Berkeley Continuous Media Toolkit Available at <<http://bmrc.berkeley.edu/frame/research/cmt/>>
- [4] Classroom 2000 Available at <<http://www.cc.gatech.edu/fce/c2000/overview/index.html/>>

---

<sup>2</sup> For example, a MANIC short course on Unix socket programming is used in half -dozen universities , with our permission.

- [5] COMPSCI 521/621 MANIC website at <<http://manic.cs.umass.edu/~cs521/>>
- [6] Dijkstra, Edsger W. On the Cruelty of Really Teaching Computer Science," *Comm. of the ACM* 32, 12 (December, 1989)
- [7] Koedinger, K., Suthers, D., and Forbus, K. Component-based Construction of a Science Learning Space, *International Journal of Artificial Intelligence in Education* 10 (1999).
- [8] MANIC Webpage at <<http://manic.cs.umass.edu/>>
- [9] MediaNet. Available at <<http://www.cs.cornell.edu/Info/Projects/MediaNet/>>.
- [10] OWL Webpage at <<http://www.cs.umass.edu/~ckc/owl/>>
- [11] Padhye J. and Kurose, J. An Empirical Study of Client Interactions with a Continuous-Media Courseware Server. *IEEE Internet Computing* (April 1999) a more detailed version of this paper appeared in *Proc. of NOSSDAV '98*, (Cambridge, UK, July 1998).
- [12] Pane, John F., Corbett, Albert T., & John, Bonnie E. Assessing Dynamics in Computer-Based Instruction *Proc. CHI 96*, (Vancouver, Canada April, 1996)
- [13] Parnas, D.L. Software Engineering Programmes are not Computer Science Programmes, CRL Report 361, Communication Research Laboratory, McMaster University (April 1998)
- [14] RealProducer Available at <<http://www.realnetworks.com/>>
- [15] Schneider, Daniel Teaching & Learning with Internet Tools: A Position Paper. Workshop on Teaching & Learning with the Web at the First International Conference on the World-Wide Web (Geneva, 1994). Also available at <<http://tecfa.unige.ch/>>
- [16] Soloway, Elliot, Jackson, Shari L., Klein, Jonathan, Quintana, Chris, Reed, James, Spitulnik, Jeff, Stratford, Steven J., Studer, Scott, Jul, Suzanne, Eng, Jim & Scala, Nancy Learning Theory in Practice: Case Studies of Learner-Centered Design. *Proc. CHI 96*, (Vancouver, Canada April, 1996)
- [17] Stern, M. K. & Woolf, B. P. Curriculum Sequencing in a Web-Based Tutor. in *Proc. of Intelligent Tutoring Systems*. (1998)
- [18] Stern, M., Steinberg, J., Lee, H.I., Padhye, J. Kurose, J. MANIC: Multimedia Asynchronous Networked Individualized Courseware in *Proc. of Educational Multimedia and Hypermedia* (1997)
- [19] Stern, M., Woolf, B. & Kurose, J. Intelligence on the Web? in *Proc. of Artificial Intelligence in Education*, (Kobe, Japan., 1997).
- [20] Wilson, B., & Cole, P. A review of cognitive teaching models. *Educational Technology Research and Development*, 39, 4 (April, 1991) Also available at: <<http://www.cudenver.edu/~bwilson>>
- [21] Wilson, Brent G., Reflections on Constructivism and Instructional Design. In *Instructional Development Paradigms*. C.R. Dills & A.A. Romiszowski, Eds. (Englewood Cliffs, NJ, March 1997)

